Global Model Management in Eclipse GMT/AM3

Freddy Allilaire, Jean Bézivin, Hugo Bruneliere, Frédéric Jouault

Global Model Management in Eclipse GMT/AM3

Freddy Allilaire, Jean Bézivin, Hugo Bruneliére, Frédéric Jouault
ATLAS (INRIA & LINA)
University of Nantes
2, rue de la Houssinière BP 92208
44322, Nantes, France
{freddy.allilaire | bezivin | hugo.bruneliere | f.jouault}@gmail.com

Keywords

Global Model Management, Megamodeling, Modeling in the Large.

Abstract

Within the GMT technology project (Generative Modeling Tools), an initial set of tools and artifacts for Global Model Management have recently been released under Eclipse. This paper presents the corresponding AM3 (ATLAS MegaModel Management) goals and contributions. AM3 is intended to provide support for modeling in the large, i.e. managing global resources in the field of MDE (Model-Driven Engineering). These global resources are usually heterogeneous and distributed. To access them without increasing the accidental complexity of MDE, we need to invent new ways to create, store, view, access, and modify the global entities that may be involved in developing a solution. To this intent, the notion of a megamodel (i.e. a model which elements are themselves models) is being used. This paper reports on the goals, the present state, and some future planned evolution of this Eclipse contribution.

1. INTRODUCTION

The idea of *megaprogramming* was introduced by B. Boehm in [13] in order to propose a solution to the construction of large-scale software systems. In [8] we have proposed the related idea of *megamodeling* in order to cope with the accidental complexity that has been observed when building real life model driven engineering (MDE) solutions to practical problems.

MDE mainly suggests basing the software development and maintenance process on chains of model transformations. A single transformation is quite easy to handle, but as soon as we tackle real life situations (for example as in [7]) we are faced with large set of MDE artifacts from which we have to assemble a solution. The classical programming level tools are of no significant help here to manage this kind of situation. If we want software systems to be designed from a high number of models, metamodels, transformations, converters and other similar components, we need to provide a development environment that will allow the user to handle this complexity without major penalties.

With such proposals as the Eclipse Modeling Framework (EMF) or the Microsoft DSL Tools being broadly used, we clearly see that MDE is going to be soon present in many organizations. Building on the ideas presented in [11] and [8] , we report in this paper on a new Eclipse prototype, named AM3 [1] for ATLAS MegaModel Management that intends to address the problems of modeling in the large. The initial set of tools is available in the GMT project (Generative Modeling Tools) which acts as a research incubator for the newly created top level modeling project.

This paper is organized as follows. Section 2 provides an overview of the theoretical aspects of our Global Model Management (GMM) approach. Section 3 gives a list of definitions for the different concepts involved in this approach. Section 4 describes the present situation in the AMMA Platform (and especially in the GMT/AM3 project) in terms of real achievements. Section 5 presents the general objectives of our work on AM3 in the domain of Global Model Management. Section 6 concludes the paper.

2. OVERVIEW

Our GMM approach is based on several general concepts. Some of them, corresponding to a conceptual MDE framework, have already been clearly identified, defined, and presented in [9] and [10]; some others are introduced here in order to address more specifically the particular problems of GMM. Figure 1 summarizes all these concepts and illustrates the way they are related. The properties concerning some of these concepts are expressed more formally, as OCL constraints, in the next section (in Figure 3 which is a zoom on Figure 1).

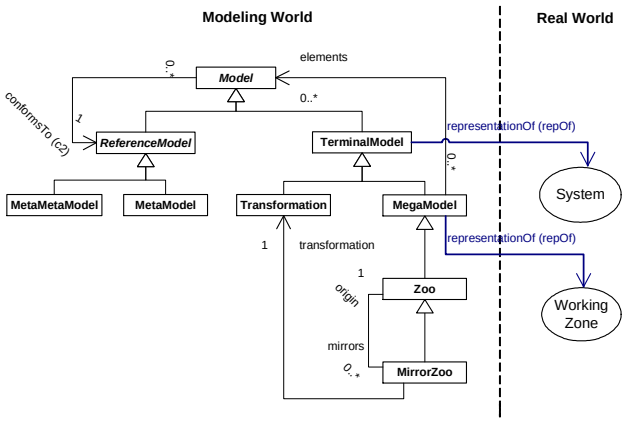

Figure 1. Theoretical aspects of our Global Model Management approach

In MDE (Model Driven Engineering), models are first class entities. Models are organized in three different categories: *terminal models*, *metamodels*, or *metametamodels*. Each model conforms to its reference model, i.e. a metamodel or a metametamodel. It is important to note that the reference model of a metametamodel is always itself. A terminal model is a

representation, that conforms to a given metamodel (its reference one) and that is also a representation of a *real world system*.

Transformations and megamodels are examples of terminal models with particular properties. A transformation is a model that can be processed (with a virtual machine for example) in order to build a target model from a source model, both of them conforming to a metamodel that can be different.

We introduce the new concept of *megamodel* [8] in order to provide some kind of registry for models in the context of GMM. A megamodel is a model that contains global entities like terminal models, metamodels, transformations etc. In our approach, a megamodel is considered as a representation of what we name a "working zone" (i.e. a context composed of various MDE artifacts). Megamodels conform to a particular type of metamodels. The basic specificities of this kind of metamodels are illustrated in Figure 2.

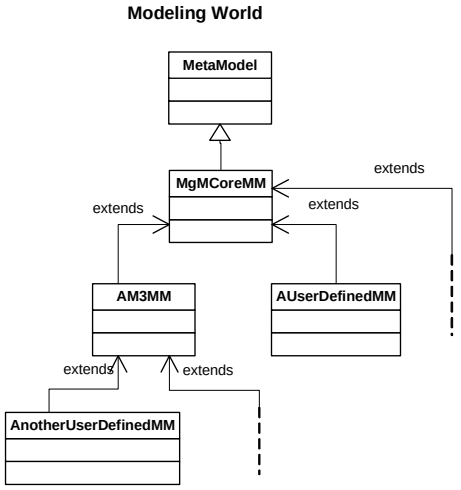

Figure 2. "Megamodels" metamodels

MgMCoreMM is the basic megamodel metamodel that defines standards resources and behaviors in the context of GMM. However users may want to refine this metamodel if they need to specify their own GMM policy. That is the reason why we introduce the "*extends*" relation in order to allow them to extend an existing metamodel by creating a new metamodel that conserves the properties of the extended metamodel.

We also add the concepts of *zoo* and *mirror zoo* which are megamodels with some specific properties. We call *zoo* a megamodel in which all elements (i.e. each model) conform to the same metamodel. As a consequence, we can imagine building zoos of terminal models, metamodels, or transformations etc. We call *mirror zoo* a zoo generated, from another zoo, by a transformation. So, each zoo may have several mirrors which conform to different metamodels.

The next section gives more precise definitions of the different concepts previously mentioned and used in this overview.

3. CONCEPTUAL DEFINITIONS

In this section, we elaborate on the concepts presented in Section 2 by giving, for each of them, a complete informal definition.

The following definitions, related to the core concept of *model*, have been taken from [10] before being reformulated:

Definition 1. A technical space [17] [12] is a model management framework, belonging to the "modeling world", with a set of tools that operate on the models definable within the framework.

Definition 2. A system is a delimited part of the world (the "real world") considered as a set of elements in interaction. It can be represented in terminal models.

Definition 3. A model is a representation of a given system. For each question of a given set of questions, the model will provide exactly the same answer that the system would have provided in answering the same question.

Definition 4. A terminal model (M1) is a model such that its reference model is a metamodel, i.e. it conforms to its reference metamodel. It is a representation of a "real world" system.

Definition 5. A metamodel (M2) is a model such that its reference model is a metametamodel, i.e. it conforms to its reference metametamodel.

Definition 6. A metametamodel (M3) is a model that is its own reference model, i.e. it conforms to itself.

Now that we have clearly defined the general concept of "model" and all its main related concepts, we must specify the new concepts we add in our approach in order to deal with the general problems of GMM:

Definition 7. A working zone is a delimited part of the world (the "real world") consisting of MDE resources.

Definition 8. A transformation is a terminal model that defines a transformation from a model $M1_A$, conforming to a source metamodel $M2_S$, to a model $M1_B$ conforming to a target metamodel $M2_T$. Its reference model is a transformation metamodel (like the metamodel of the ATL language for example [5]).

Definition 9. A megamodel is a terminal model such that all its elements are models (i.e. all kinds of modeling artifacts and modeling tools like terminal models, metamodels, metametamodels…). It is a representation of a "real world" working zone. We consider this concept as the core of our GMM approach but also as a central part of the "modeling in the large" principle discussed in [11].

Definition 10. A zoo is a megamodel such that all models that compose it have the same metamodel (i.e. the same reference model). The kind of modeling artifact that can be found in a zoo may vary (as an example, the "Atlantic zoo" and the "ATL transformation zoo" which are located in [1] are zoos, respectively of metamodels and of transformations). Alternatively, a zoo may be considered as a view on a megamodel. This view may be implemented as a transformation for example. A zoo may have several mirrors, each of them having this zoo as original zoo.

Definition 11. A mirror zoo is a zoo that has been automatically generated, by the execution of a given transformation, from a specified original zoo. There are several types of events that can be the triggers of the generation, or regeneration, of a mirror zoo (as an example, when a modification occurs in the original zoo…).

All the concepts detailed in this section are summarized in Figure 1. As it has been specified in the previous definitions, some of them have particular properties. These properties can be expressed as OCL constraints such as those presented in Figure 3.

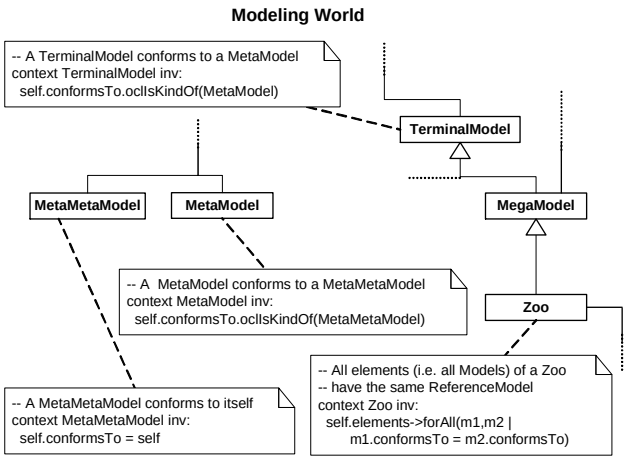

Figure 3. OCL constraints on some GMM general concepts (zoom on Figure 1)

The next section describes the present situation, in the AMMA Platform, in terms of Global Model Management.

4. THE PRESENT SITUATION

In the AMMA Platform, our Global Model Management approach is materialized by the AM3 (ATLAS MegaModel Management) Eclipse/GMT subproject [1]. The goal of this project is to provide a global environment for modeling in the large, i.e. dealing with all kinds of modeling artifacts in the context of an interoperability and open platform.

A preliminary original description of the AM3 project in general terms (its presentation, its motivations, its use cases, and its global objectives) has already been given in [11]. However the content of the AM3 project has much evolved during the recent period. In fact, this project currently offers to Eclipse community's users some general documentations, three functional free-downloadable plugins and two zoos (with several mirrors for one of the two zoos), as shown in Figure 4.

4.1 Current Functionalities and Work

The current implementation of the AM3 project provides many basic features in the context of GMM. In fact, it already allows users to handle the most common kinds of modeling artifacts (i.e. most common resources and their associated tools).

A default *megamodel metamodel* and a basic AM3 megamodel (conforming to this metamodel) are currently defined and have been contributed to AM3. This available megamodel involves only local resources for the time being. It can be interrogated by OCL requests in order to retrieve some particular resources such as models, metamodels, editors, outlines, projectors (injectors or extractors), etc. In addition to that, it is also possible to add new artifacts to this megamodel and to specify their metadata. Users can define their own megamodels representing their own working zone by using the TCS (Textual Concrete Syntax) provided in AM3 to facilitate the creation of megamodels. Currently, there is only one megamodel loaded in memory at the same time: this megamodel is a merge of all the available megamodels (in the workspace and in the plugins).

Figure 4. The AM3 Project home page [1]

Resources found in a megamodel can be directly handled with AM3 or with the other main tools of the AMMA Platform (ATL [5] and AMW [6]). Furthermore, AM3 will soon offer the possibility to dynamically request metamodels and models by using OCL queries (the development is currently in process and first results had already been observed).

Finally, in order to synthesize the basic resources and tools provided with the AMMA Platform (KM3, ATL, etc), we are currently gathering them into a specific package that we call *AMMACore*. A first preliminary version of this package will be soon available with the next AM3 version.

4.2 Zoos (Resources)

As previously mentioned, the AM3 project already provides many MDE resources which are mainly at this time, metamodels and transformations (as it is shown in Figure 5).

Figure 5. The AM3 Zoos page

In fact, three zoos and several mirrors are currently freely available in the AM3 project:

Zoo 1. The Atlantic Zoo: It is composed of metamodels expressed in KM3 (Kernel MetaMetaModel) format [9]. This zoo has several mirrors that are auto-generated from it by using model transformations in ATL (the number of mirrors is variable and

evolving, with new mirror zoos forthcoming like one in the Prolog language):

- **Mirror 1.** The AtlantEcore Zoo: It contains metamodels expressed in EMF **XMI 2.0**, conforming to **Ecore**.
- **Mirror 2.** The Atlantic MOF/MDR Zoo: It contains metamodels expressed in MDR **XMI 1.2**, conforming to **MOF 1.4**.
- **Mirror 3.** The Atlantic UML Zoo: It contains UML - class diagram's representations of metamodels expressed in MDR **XMI 1.2**, conforming to **UML**, that are compatible with the Poseidon UML CASE tool [15].
- **Mirror 4**. The Atlantic Raster Zoo: It contains graphical representations of metamodels expressed in **PNG** bitmaps.
- **Mirror 5**. The Atlantic SQL DDL Zoo: It contains metamodels' representations expressed in **SQL DDL** (Data Definition Language), conforming to **SQL**. They have been tested with the MySQL DBMS [19].
- **Mirror 6**. The Atlantic Microsoft DSL Tools Zoo: It contains "domain models" expressed in the **DSL Tools** specific XML format (".dsldm" files). These files are usable under Visual Studio 2005 with the Visual Studio 2005 SDK (including the DSL Tools) [7] [14].
- **Mirror 7**. The Atlantic Microsoft Visual Basic Zoo: It contains metamodels expressed in Visual Basic source code for Visual Studio 2005 [18].
- **Mirror 8**. The Atlantic XAMS Zoo: It contains metamodels expressed as abstract machines in XAMS which is an open source compiler for Abstract State Machines (ASMs) [3].
- **Mirror 9**. The Atlantic AsmL Zoo: It contains metamodels expressed as abstract state machines in the Microsoft Abstract State Machine Language [4].
- **Mirror 10**. The Atlantic GME Zoo: It contains metamodels expressed in the Generic Modeling Environment (GME) format [16].

Zoo 2. The ATL Transformation Zoo: It is composed of model transformation expressed in ATL (ATLAS Transformation Language) format [5].

Zoo 3. The AMW Zoo: It is composed of weaving metamodels and extensions of weaving metamodels expressed in KM3 format. For the moment, it provides the AMW weaving core metamodel and some already defined basic extensions metamodels.

All files contained in these zoos or mirror zoos are directly downloadable in the "Zoos" page of the AM3 Project. Several other mirrors of the Atlantic Zoo, as well as mirrors of the ATL Transformation Zoo, may be created and added to the list of available zoos in this project.

4.3 Plugins (Tools Implementation)

The AM3 project also provides, in addition to the MDE resources described in section 4.2, a set of tools that allow users to handle theses kinds of resources. These tools currently implement the functionalities previously presented in section 4.1. In a more technical point of view, these tools are encapsulated into three different Eclipse plugins:

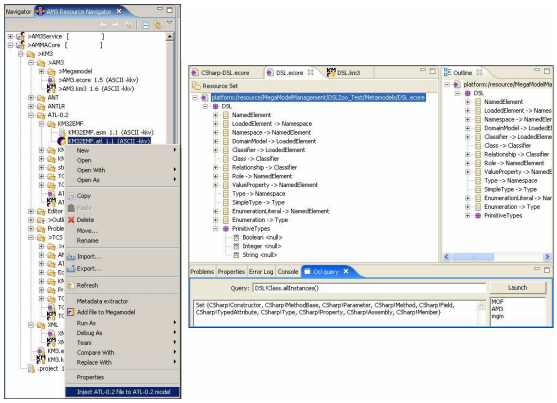

Figure 6. "AM3 Resource Navigator" & "OCL Query View"

Plugin 1. org.eclipse.am3.core: The Core plugin provides resources and the general mechanism (of the megamodel) used to handle and manage these resources in our GMM environment. For the moment, a metamodel of megamodels is defined and loaded by this plugin as well as a basic AM3 megamodel that conforms to this metamodel.

Note that the *AMMACore* project (mentioned in 4.1), which is still in development, already provides a TCS (Textual Concrete Syntax) for megamodels definitions.

Plugin 2. org.eclipse.am3.ui: The UI (User Interface) plugin provides a user interface under Eclipse for the Core plugin. The already implemented "AM3 Resource Navigator" and the still in development "OCL Query View" (see Figure 6) are typical examples of that kind of UI. This user interface will be progressively improved by adding new tools and new functionalities.

Plugin 3. org.eclipse.am3.tools.tge: The TGE (Textual Generic Editor) plugin provides an editor and outline for textual languages. To provide these two features, TGE needs the metamodel of the language, a model conforms to the Generic Editor metamodel, a model conforms to the Generic Outline metamodel and an injector. Figure 7 provides an example of use for the TGE plugin, as it is already implemented in AM3 for the KM3 language.

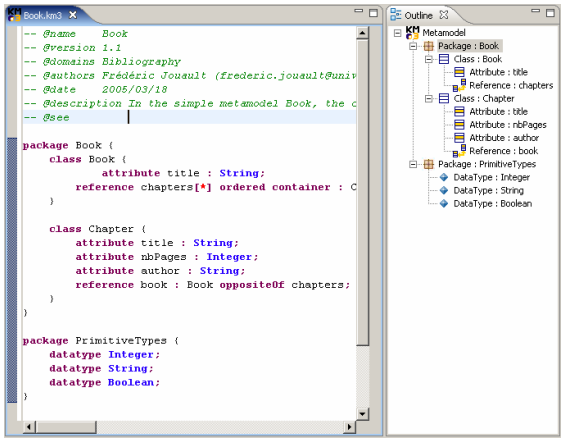

Figure 7. TGE: an example of use

These three plugins, although being already functional, will undergo significant evolution (in terms of available functionalities) in the next months.

5. OBJECTIVE OF THE WORK

The purpose of the future AM3 project evolutions is to provide an environment for GMM that respects the general abstract concepts discussed in section 2 and 3, and that gives concrete answers to the needs expressed throughout this paper (and throughout other papers such as [11] and [8]).

Our main objective is to be able to handle, with AM3, the totality of the AMMA Platform tools and resources but also to be able to import and use external ones. Figure 8 summarizes this objective by showing our future vision of the AM3 project in terms of components and global organization.

In this context, a megamodel will be associated to each MDE project, under the AMMA Platform, in order to define the project's GMM policy (i.e. define the artifacts and their location as well as the actions associated to each kind of resource, tool, etc). The megamodel will have to be easily navigable, so we will have to provide an advanced search and filtering tool.

The MDE artifacts (internal ones or imported external ones) may be stored locally or geographically distributed. As a consequence, several synchronization mechanisms will probably have to be designed and implemented in order to solve this kind of problems. Moreover, we want to automate to the maximum the actions' chains in which MDE artifacts are involved. Thus, ANT tasks [2] for ATL are currently being implemented in order to allow the building of complex ANT scripts for complex chains of transformations. One of our goals is to extend this principle to other kinds of modeling artifacts and to provide an advanced user interface to facilitate the building of such scripts (Figure 8).

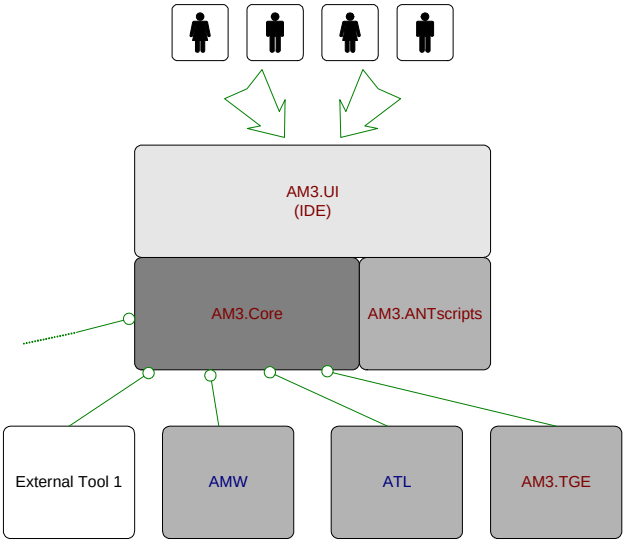

Figure 8. A future vision of AM3

In addition to all the previously mentioned objectives, our wish is also to increase as much as possible the number of MDE resources available in the AM3 project by continuously creating and adding new zoos, new mirrors zoos, etc.

6. CONCLUSIONS

We have reported in this paper on the recent advances of an Eclipse project named AM3 and dealing with GMM. Although the contributed tools are still in initial state, it is possible to see now the initial vision concretely taking shape.

We are working in an area where we need experimental and iterative investigations. When the Smalltalk team at PARC designed the Smalltalk class browser to handle a library of about 300 classes, they had to design, experiment, evaluate, and design again. In this area of GMM it is very similar. We know that MDE does not mean that you are going to translate UML to Java in one-shot operations. We have discovered in our first real-life experiments that the situation is much more tricky and complex. We need to offer to the MDE engineer an easy and convenient access to all the resources he/she may need to builds a final solution. The tools that AM3 is offering today are going some distance in this direction, but need still to be completed and enhanced. Of course building on a sound conceptual foundation as has been described in Section 3 is also an essential requirement for this iterative experimental work to be able to converge.

7. ACKNOWLEDGEMENTS

We would like to thank all the members of the ATLAS team for their help in this ongoing work, and specially Patrick Valduriez, Ivan Kurtev and Guillaume Hillairet. This work has been partially supported by ModelWare, IST European project 511731.

8. REFERENCES

[1] **AM3 "ATLAS MegaModel Management"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/am3/

[2] **ANT**, **The Apache Ant Project**: http://ant.apache.org/

[3] **ASM (Abstract State Machine)**, A Formal Method for Specification and Verification: http://www.eecs.umich.edu/gasm/

[4] **AsmL (Microsoft Abstract State Machine Language)** official site: http://research.microsoft.com/fse/asml/

[5] **ATL "ATLAS Transformation Language"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/atl/

[6] **AMW "ATLAS Model Weaver"** official site (**Eclipse/GMT** subproject): http://www.eclipse.org/gmt/amw

[7] Bézivin, J., Hillairet, G., Jouault, F., Kurtev, I., and Piers, W., **Bridging the MS/DSL Tools and the Eclipse Modeling Framework**. In: Proceedings of the International Workshop on Software Factories at OOPSLA 2005, San Diego, California, USA.

[8] Bézivin, J., Jouault, F., and Valduriez, P., **On the Need for Megamodels**. In: Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications.

[9] Bézivin, J., Jouault, F., **KM3: a DSL for Metamodel Specification**. In: Proceedings of 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems, Bologna, Italy.

[10] Bézivin, J., Jouault, F., Kurtev, I., and Valduriez, P., **Model-Based DSL Frameworks**. Submitted for Publication.

[11] Bézivin, J., Jouault, F., Rosenthal, P., and Valduriez, P., **Modeling in the Large and Modeling in the Small**. In: Proceedings of the European MDA Workshops: Foundations and Applications, MDAFA 2003 and MDAFA 2004, LNCS 3599, edited by Uwe Aßmann, Mehmet Aksit, Arend Rensink. Springer-Verlag GmbH, pages 33-46.

[12] Bezivin, J., Kurtev, I., **Model-based Technology Integration with the Technical Space Concept**. In: Metainformatics Symposium 2005, Esbjerg, Denmark, November 2005, to be published in LNCS volume

[13] Boehm, B. Sherlis, W. **MegaProgramming,** in Proc. of the DARPA Software Technology Conference, (Arlington, Va, April 1992.)

[14] **DSL Tools** for **Visual Studio 2005**, **Microsoft** official site: http://msdn.microsoft.com/vstudio/dsltools/default.aspx

[15] Gentleware **Poseidon for UML** (UML Case Tool), Gentleware official site: http://gentleware.com/index.php

[16] **Generic Modeling Environment (GME)** official site: http://www.isis.vanderbilt.edu/projects/gme/

[17] Kurtev, I., Bézivin, J., Aksit, M., **Technical Spaces: An Initial Appraisal**. CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002 http://www.sciences.univ-nantes.fr/lina/atl/publications/

[18] Microsoft**, Visual Basic 2005**, Microsoft official french site: http://www.microsoft.com/france/msdn/vbasic/default.mspx

[19] **MySQL** (DBMS), official site: http://www.mysql.com